DIP 5th Lecture

September 19, 2023

1 Lecture 5: Digital Image Processing

Distance Formula: Ececlidean Distance, Block Distance or Manhattan distance and, Chessboard Distance or Chebyshev distance

1.1 Objectives

- To learn about the distance formula
- To learn about the Euclidean distance
- To learn about the Block distance or Manhattan distance
- To learn about the Chessboard distance or Chebyshev distance
- Using the distance formula to find the distance between two pixels
- Using Ecledian distance to draw a circle
- Using Block distance to draw a box
- Using Chessboard distance to draw a diamond

1.1.1 Importing the required libraries

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

1.1.2 Creating a Binary Image

Draw an image of size 300x300 with all pixels having value 1 and having two points of value 0 at (100,100) and (200,200).

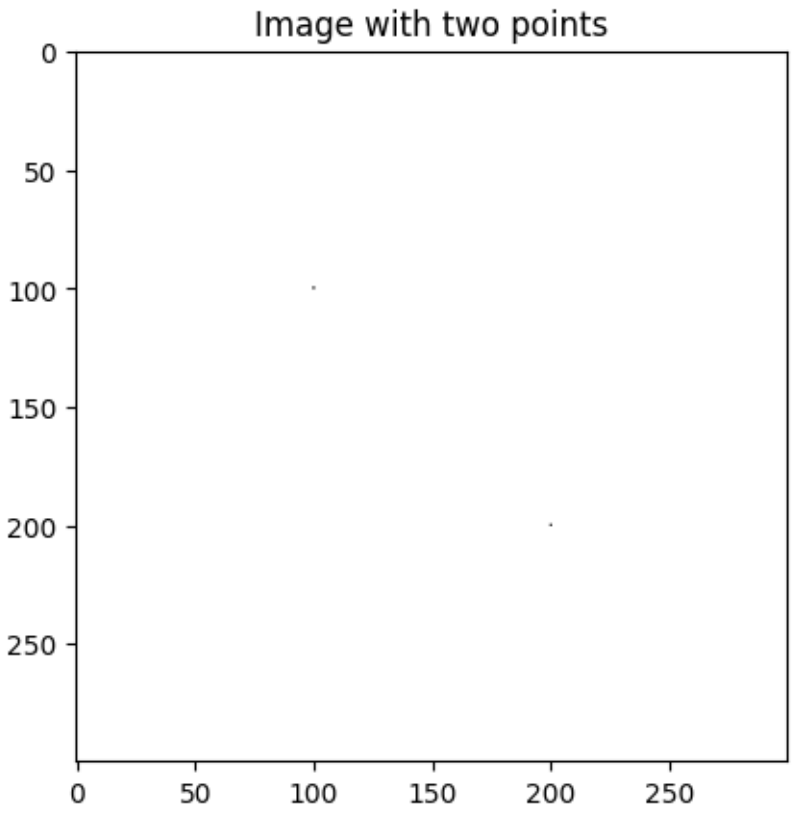
[ ]:
```
image = np.ones((300, 300))

p1 = [100, 100]
p2 = [200, 200]

image[p1[0], p1[1]] = 0
image[p2[0], p2[1]] = 0

plt.title("Image with two points")
plt.imshow(image, cmap="gray")
```

[ ]: <matplotlib.image.AxesImage at 0x223ce087e90>



1.1.3 Euclidean Distance

Euclidean distance, also known as L2 distance or Euclidean norm, is a measure of the straight-line distance between two points in Euclidean space.

Formula: Euclidean Distance = √((x2 - x1)^2 + (y2 - y1)^2)

1.1.4 Manhattan Distance (City Block Distance)

Manhattan distance, also known as L1 distance or Taxicab distance, measures the distance between two points by summing the absolute differences of their coordinates along each axis.

Formula: Manhattan Distance = |x2 - x1| + |y2 - y1|

1.1.5 Chessboard Distance (Chebyshev Distance)

Chessboard distance, also known as L∞ distance, measures the maximum absolute difference between coordinates among all dimensions.

Formula: Chessboard Distance = max(|x2 - x1|, |y2 - y1|)

1.1.6 Implementing these Distance Formula as Function

[ ]:
```
def euclidean_distance(p1, p2):
    return math.sqrt(((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2))
```

[ ]:
```
def cityblock_distance(p1, p2):
    return abs(p2[0] - p1[0]) + abs(p2[1] - p1[1])
```

[ ]:
```
def chessboard_distance(p1, p2):
    return max([abs(p2[0] - p1[0]), abs(p2[1] - p1[1])])
```

[ ]:
```
print(euclidean_distance(p1, p2))
print(cityblock_distance(p1, p2))
print(chessboard_distance(p1, p2))
```

```
141.4213562373095
200
100
```

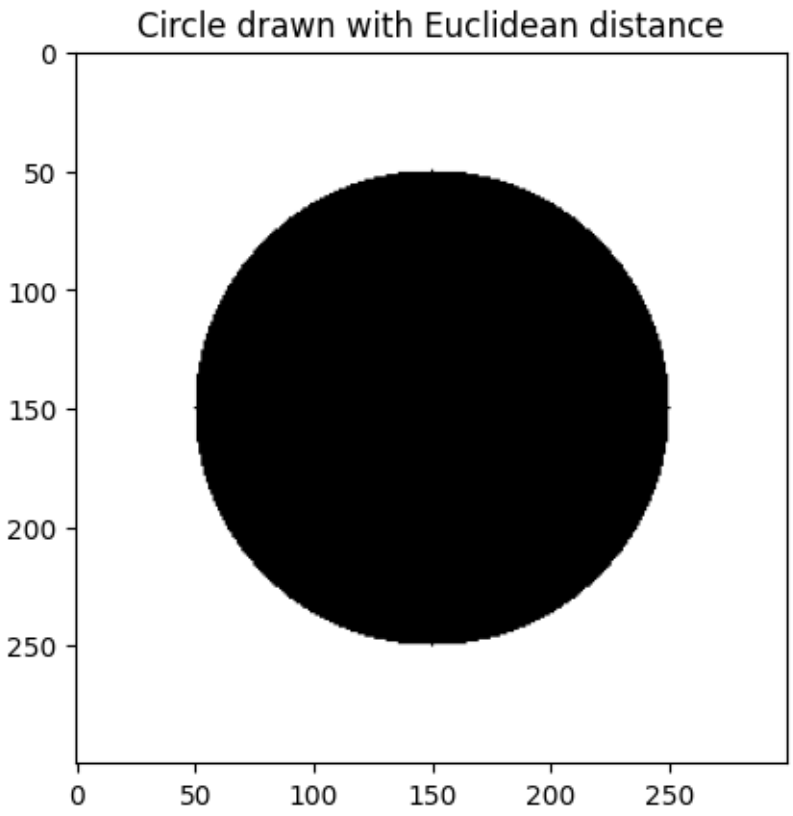
1.1.7 Drawing a Circle using Euclidean Distance

Draw a circle with center (x, y) and radius r in a binary image using Euclidean distance. The circle is approximated by a polygon of n sides. The function returns a binary image with the circle drawn.

[ ]:
```
def draw_circle(center, radius):
    image_circle = image.copy()
    for M in range(300):
        for N in range(300):
            dist = euclidean_distance(center, [M, N])
            if dist <= radius:
                image_circle[M, N] = 0
    return image_circle
```

[ ]:
```
image_circle = draw_circle([150, 150], 100)
plt.title("Circle drawn with Euclidean distance")
plt.imshow(image_circle, cmap="gray")
```

[ ]:
```
<matplotlib.image.AxesImage at 0x223cac61b50>
```



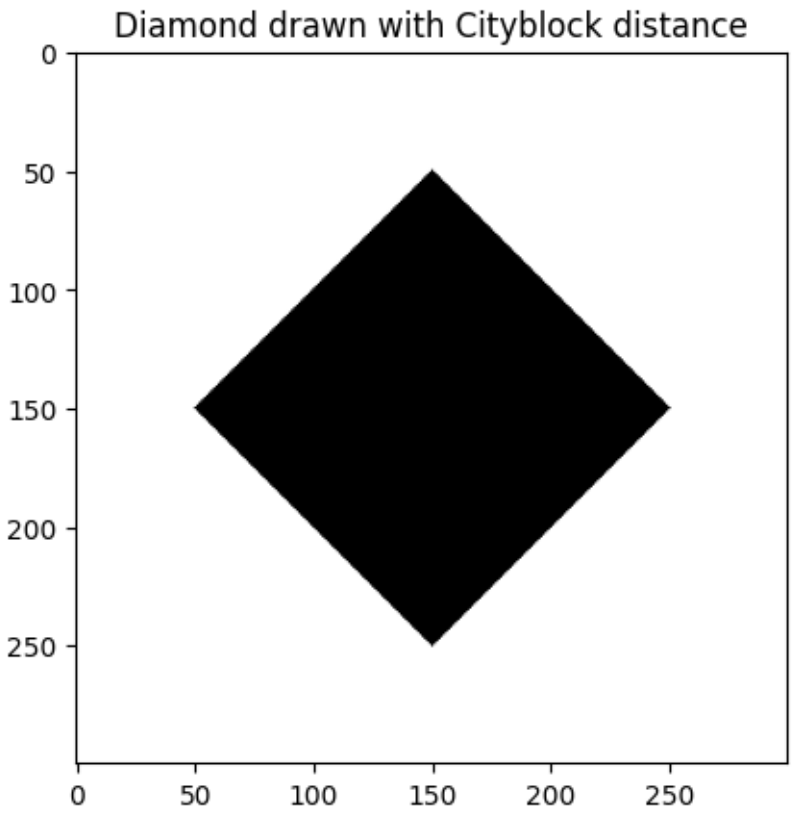
1.1.8 Drawing a Diamond using Chessboard Distance

Draw a diamond with center (x, y) and radius r in a binary image using Chessboard distance. The diamond is approximated by a polygon of n sides. The function returns a binary image with the diamond drawn.

[ ]:
```
def draw_diamond(center, radius):
    image_diamond = image.copy()
    for M in range(300):
        for N in range(300):
            dist = cityblock_distance(center, [M, N])
            if dist <= radius:
                image_diamond[M, N] = 0
    return image_diamond
```

[ ]:
```
image_diamond = draw_diamond([150, 150], 100)
plt.title("Diamond drawn with Cityblock distance")
plt.imshow(image_diamond, cmap="gray")
```

[ ]:
```
<matplotlib.image.AxesImage at 0x223cdf8fbd0>
```



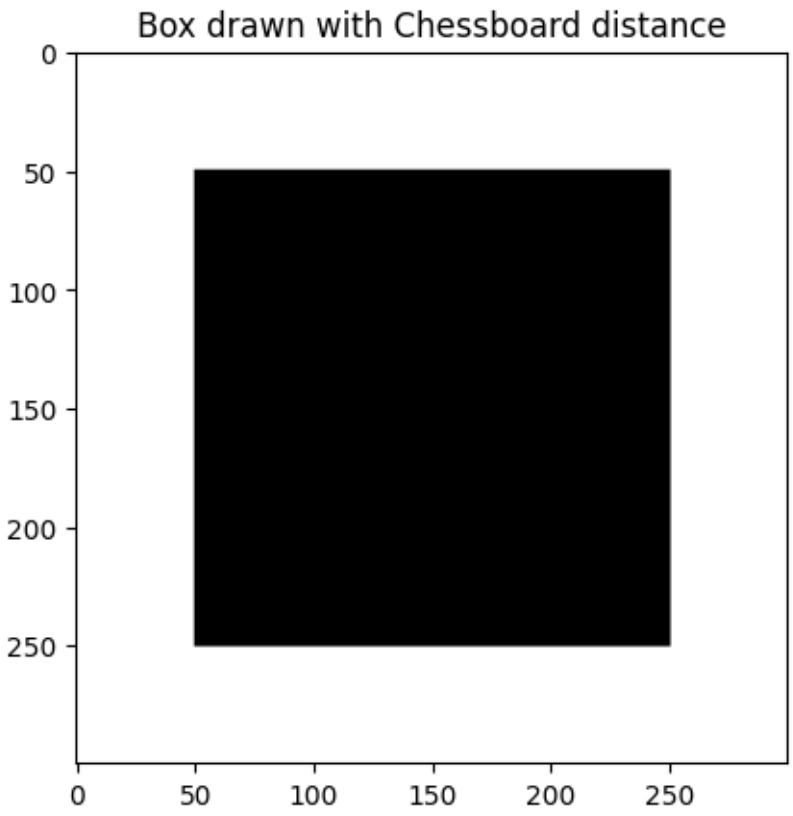
1.1.9 Drawing a Box using Manhattan Distance

Draw a box with center (x, y) and radius r in a binary image using Manhattan distance. The box is approximated by a polygon of n sides. The function returns a binary image with the box drawn.

[ ]:
```
def draw_box(center, radius):
    image_box = image.copy()
    for M in range(300):
        for N in range(300):
            dist = chessboard_distance(center, [M, N])
            if dist <= radius:
                image_box[M, N] = 0
    return image_box
```

[ ]:
```
image_box = draw_box([150, 150], 100)
plt.title("Box drawn with Chessboard distance")
plt.imshow(image_box, cmap="gray")
```

[ ]:
```
<matplotlib.image.AxesImage at 0x223cdffca10>
```



1.2 Conclusion

Congratulations! You have successfully implemented the distance formula and used it to draw a circle, diamond and box. We can also use these distances as a feature matrix for classification. In other words this is the simplest way to classify the images.